PCA For Taxi Duration Prediction

Table of Contents

Objectives 1
Brief description of the data set and a summary of its attributes 1
Data Exploration 3
Data Cleaning and Feature engineering 3
Data Preparation for Machine Learning 5
Target Variable 5
Feature Selection 5
Feature transformation 6
Model Training and Evaluation 7
Key Findings and Insights 11
Model Selection 11
Next Steps in model improvement 15
Summary 15

Objectives

In an earlier report, our best model for predicting trip duration in minutes used polynomial features which increased the number of features used in making our prediction.

Here the main objective is dimensionality reduction using principal component analysis (PCA) technique to reduce the number of features before adding polynomial features. This approach will reduce the size of our model thereby speeding up training and even achieving similar accuracy with our best model without PCA.

Brief description of the data set and a summary of its attributes

TLC Trip Record Data has 12 years (2009 –2020) worth of Data available but I have decided to analyze a subset of the 2015.

In 2015, passengers took nearly 300 million yellow cab rides in New York City. Working with the complete dataset for all these rides would require considerable time and computational resources. The 12 data files used represent two percent of the total trips sampled at random from each month.

I chose this data because it is useful for real-world applications such as:

- Predicting taxi duration for a trip
- Allocating taxi to zones/regions based on demand.

Below is the summary of the data attributes as described here

Attribute / Column	Description
VendorID	A code indicating the TPEP provider that provided the record. **1= Creative Mobile Technologies, LLC; 2= VeriFone Inc.**
tpep_pickup_datetime	The date and time when the meter was engaged.
tpep_dropoff_datetime	The date and time when the meter was disengaged.
Passenger_count	The number of passengers in the vehicle. This is a driver-entered value.
Trip_distance	The elapsed trip distance in miles reported by the taximeter.
PULocationID	TLC Taxi Zone in which the taximeter was engaged.
DOLocationID	TLC Taxi Zone in which the taximeter was disengaged.
RateCodeID	The final rate code in effect at the end of the trip. **1= Standard rate** **2=JFK** **3=Newark** **4=Nassau or Westchester** **5=Negotiated fare** **6=Group ride**
Store_and_fwd_flag	This flag indicates whether the trip record was held in vehicle memory before sending to the vendor, aka "store and forward," because the vehicle did not have a connection to the server. **Y= store and forward trip** **N= not a store and forward trip**
Payment_type	A numeric code signifying how the passenger paid for the trip. **1= Credit card** **2= Cash** **3= No charge** **4= Dispute** **5= Unknown** **6= Voided trip**
Fare_amount	The time-and-distance fare calculated by the meter.
Extra	Miscellaneous extras and surcharges. Currently, this only includes the \$0.50 and \$1 rush hour and overnight charges.

MTA_tax	$0.50 MTA tax that is automatically triggered based on the metered rate in use.
Improvement_surcharge	$0.30 improvement surcharge assessed trips at the flag drop. The improvement surcharge began being levied in 2015.
Tip_amount	Tip amount –This field is automatically populated for credit card tips. Cash tips are not included.
Tolls_amount	Total amount of all tolls paid in trip.
Total_amount	The total amount charged to passengers. Does not include cash tips.

Data Exploration

Joining the 12 files provided resulted in 2, 922, 266 instances in the dataset with no missing values. We have 12 numerical features, 4 categorical features, 2 datetime features and 1 integer. For easy visualization, I converted *VendorID, RateCodeID and payment_type* to their corresponding values.

Summary of some the numerical attributes shows that this data is not perfect and therefore needs some cleaning for example there is no way we can have a negative tip or zero passengers.

	Passenger_count	Trip_distance	Fare_amount	Tip_amount
min	0.0	0.0	-150.00	-2.7
max	9.0	14680110.0	410266.86	650

Data Cleaning and Feature engineering

For data preprocessing/cleaning

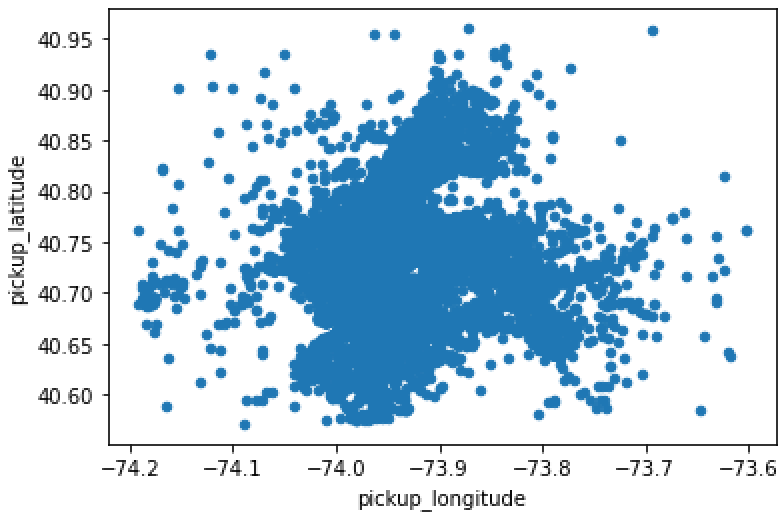
- Charges and trip information that are negative, as well as charges inconsistent with expected values are considered incorrect. In addition, trips with pickup or drop off locations outside a geographic region of interest are removed.
- Only keep trips with valid passenger and distance information.
- Remove trips missing valid pickup or drop off locations.
- Remove trips with invalid Rate code.

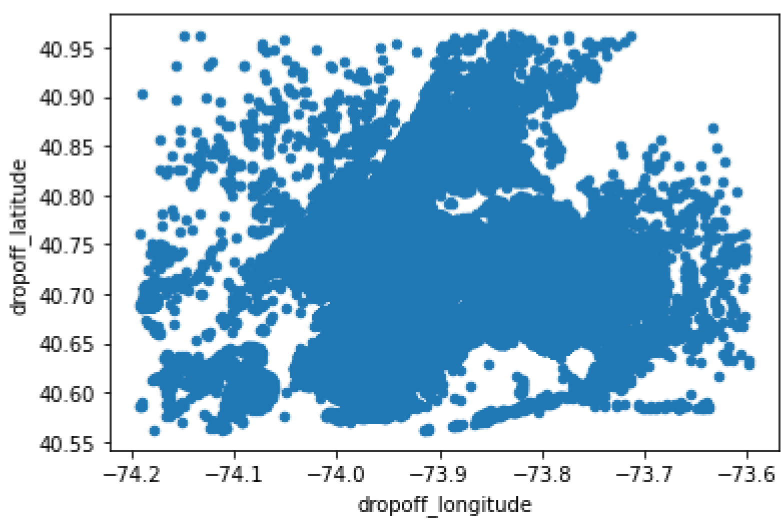
For Feature engineering, the following features were added.

- ***duration*** - Length of the trip, in minutes, calculated from the pickup and drop off times.

- ***avespeed*** **-** Average speed, in mph, calculated from the distance and duration values.
- ***time_of_day*** **-** This feature represents pick up time as the elapsed time since midnight in decimal hours (e.g. 7:10 am becomes 7.1667). The output is a duration vector with units of hours.
- ***day_of_week*** **-** This feature is a categorical array indicating the day of the week the trip began, in long format (e.g. 'Monday').

After removing invalid trip information, we can now visualize the features again.

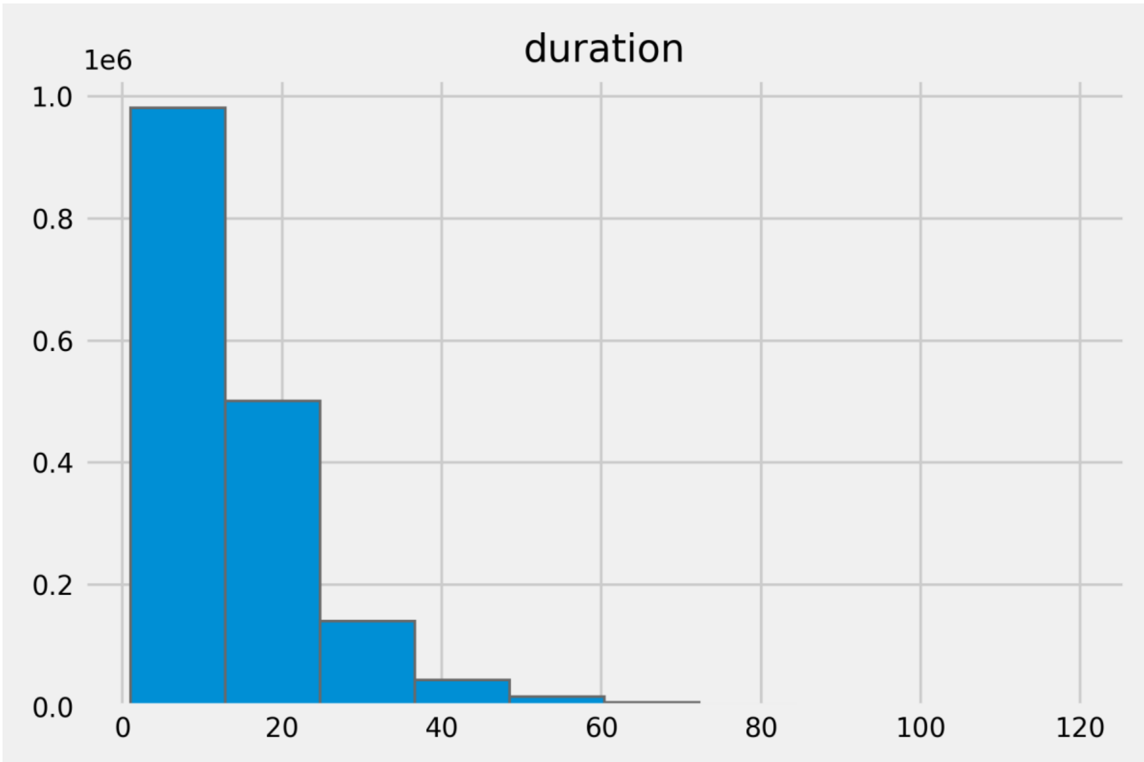




Data Preparation for Machine Learning

Target Variable

We chose the target variable to be ***duration*** as we want to predict trip duration.



Feature Selection

To avoid using all the 22 columns as features, we tried to find the correlation between each numerical feature and the target variable, and the result is as below:

Features	Correlation with duration
fare_amount	**0.888203**
total_amount	**0.862494**
trip_distance	**0.781401**
tip_amount	**0.493078**
tolls_amount	**0.457660**
pickup_longitude	**0.370568**
dropoff_longitude	**0.266498**
ave_speed	**0.159938**
time_of_day	**0.032376**
passenger_count	0.015561
extra	**-0.055389**
dropoff_latitude	**-0.179526**
pickup_latitude	**-0.233615**

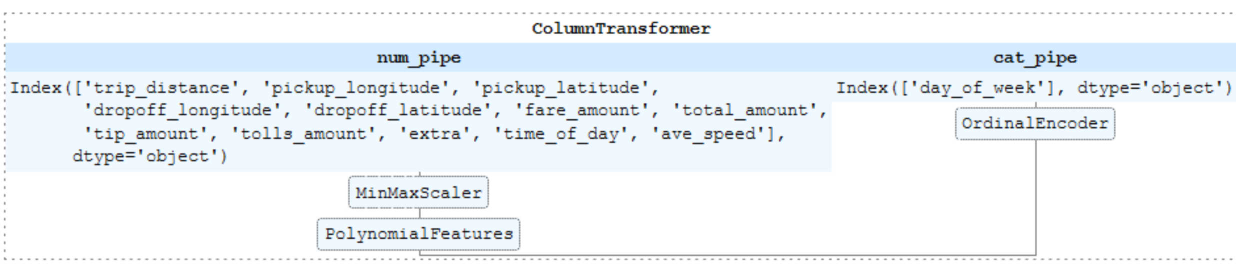
We opted to use the highlighted numerical features and ***day_of_week*** as the only categorical feature.

Feature transformation

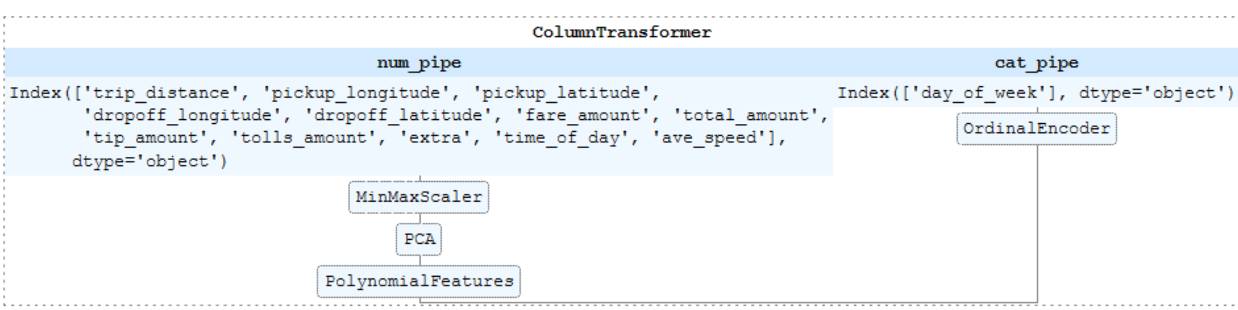
We applied two general transformations on the features: feature scaling for the numerical features and ordinal encoding for the categorical feature.

We built two transformation pipelines using scikit-learn's ***Pipeline*** and C***olumnTransformer*** classes.

The first pipeline added only polynomial features to the numerical features as show below.



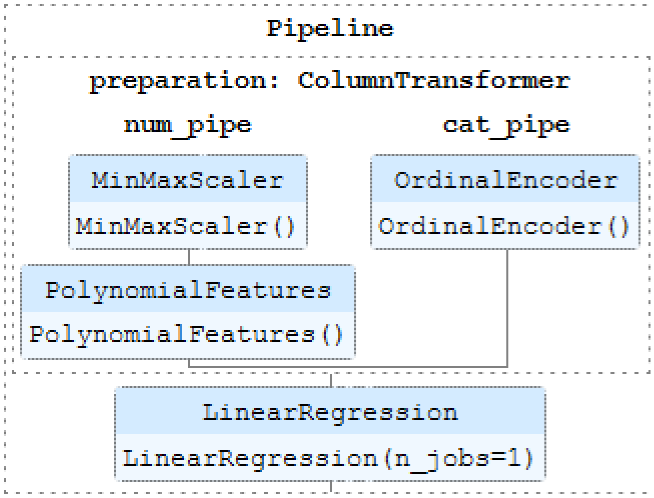
We also built a second pipeline which uses PCA to reduce the number of features before adding polynomial features.



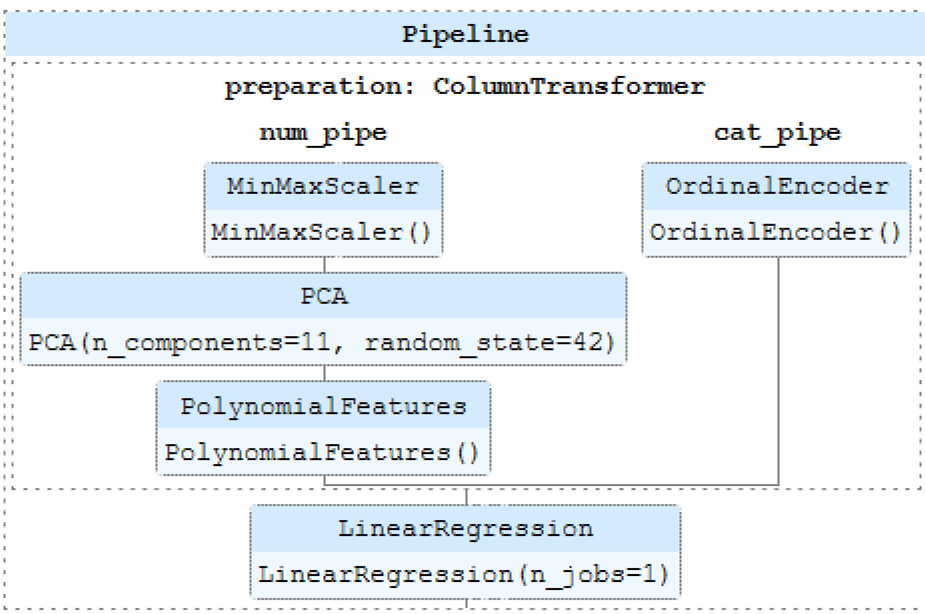
Model Training and Evaluation

We trained 7 different regression models using ***GridSearchCV*** to select the best parameters for each model and the models are shown below:

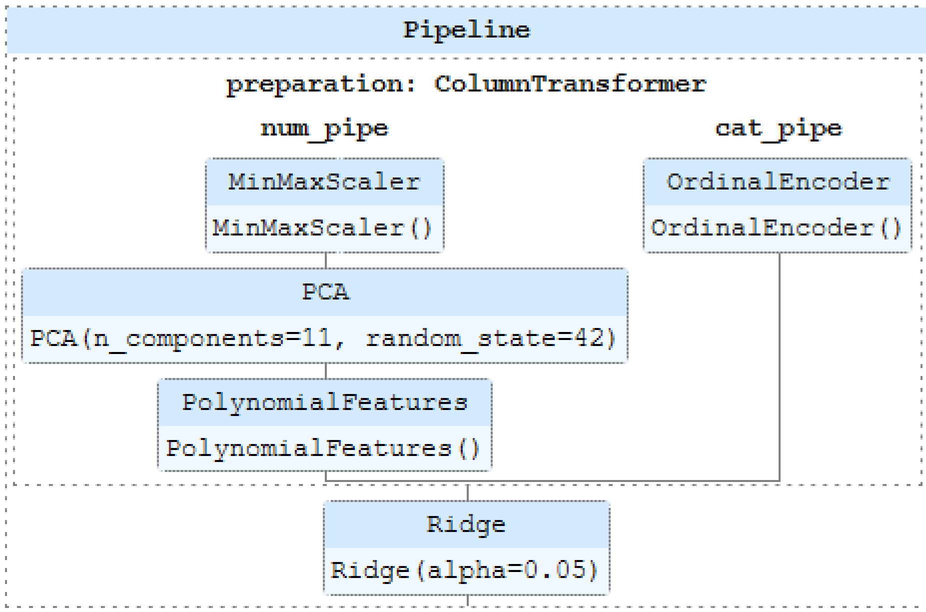
- Base Linear Regression model (no PCA)



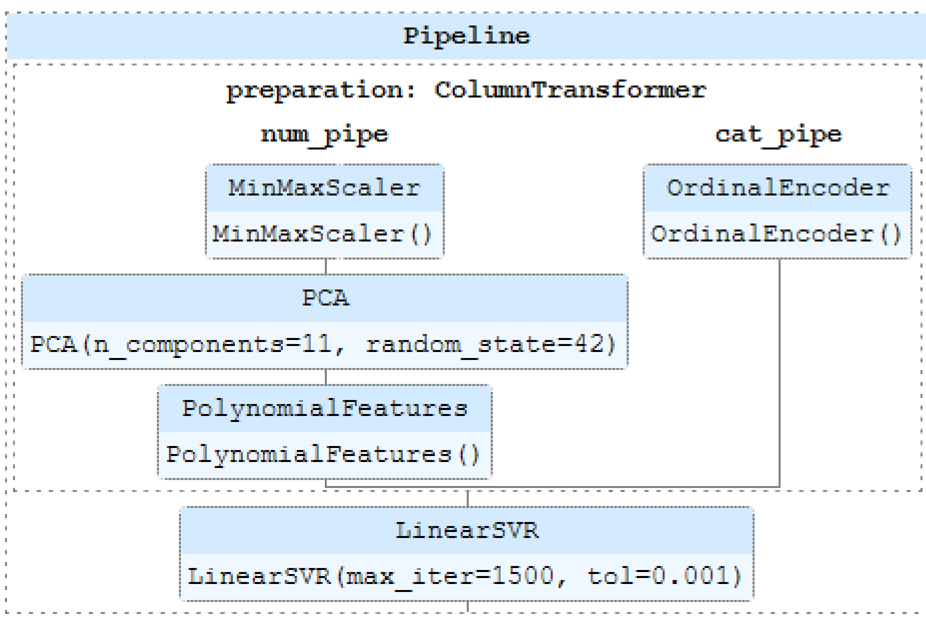
- Linear Regression model with PCA



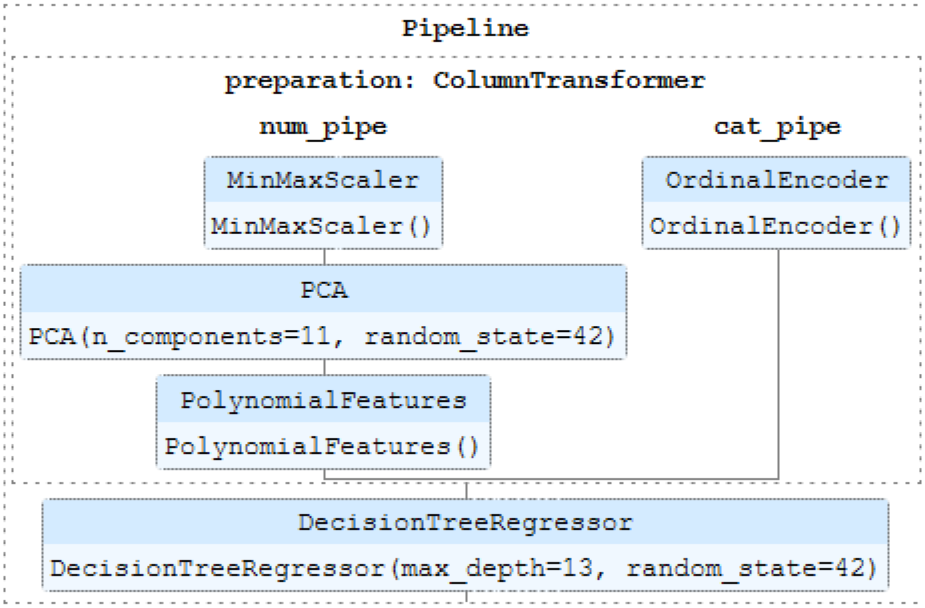
- Ridge model



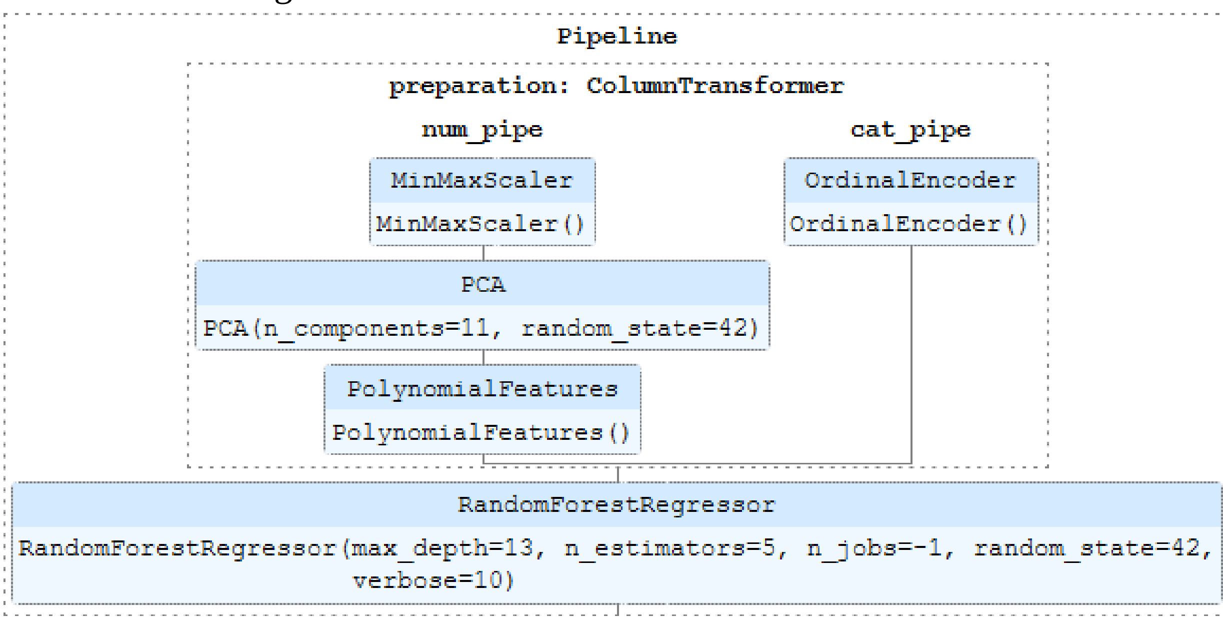
- Linear Support Vector Regressor (LinearSVR)



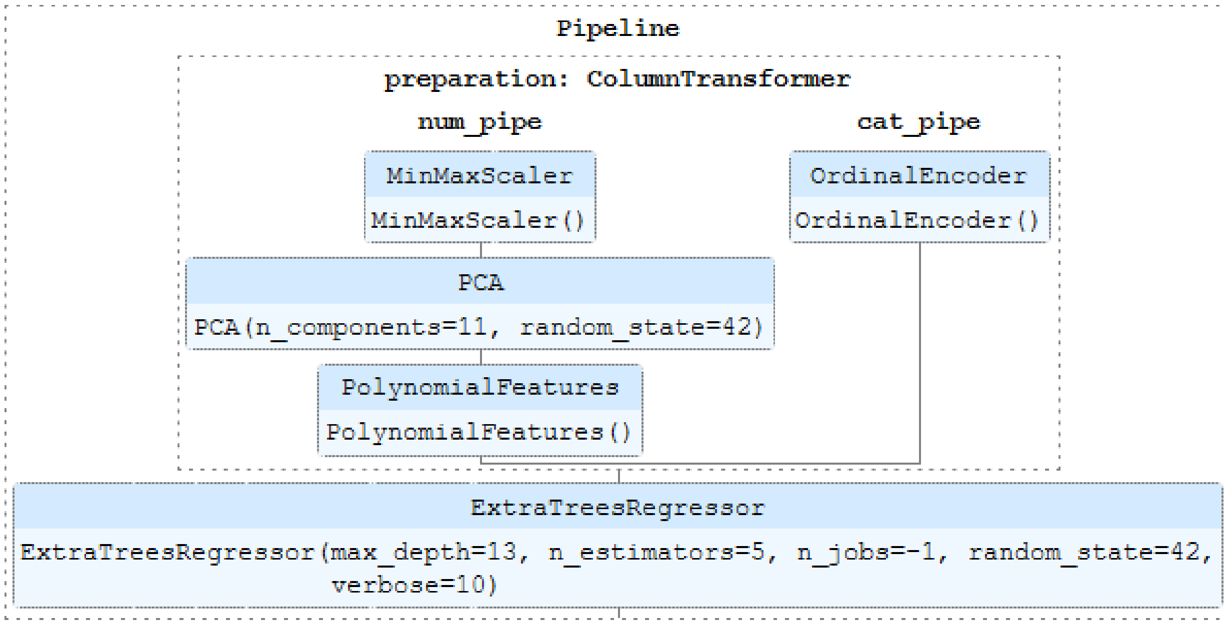
- Decision Tree Regressor



- Random Forest Regressor



- Extra Tree Regressor



We split our data into three: train (1,694,220 rows), validation (677,688 rows) and test (451,793 rows).

We evaluated the model using r2_score and rmse and obtain below outputs:

	Linear Regression (W/o PCA)		Linear Regression		Ridge Regression		LinearSVR	
	Score	Rmse	Score	Rmse	Score	Rmse	Score	Rmse
Train	0.975362	1.688281	0.974740	1.688282	0.974733	1.688340	0.965703	1.946387
Validation	0.975203	1.695001	0.974538	1.695001	0.974533	1.695027	0.965852	1.941794
Test	0.975240	1.692464	0.974598	1.692464	0.974583	1.692832	0.965296	1.957936

	Decision Tree		Random Forest		Extra Tree	
	Score	Rmse	Score	Rmse	Score	Rmse
Train	0.974485	1.696562	0.981121	1.449027	**0.977820**	**1.545799**
Validation	0.962040	2.068840	0.973969	1.698813	**0.970822**	**1.767635**
Test	0.962483	2.055130	0.974588	1.677891	**0.971184**	**1.755405**

Key Findings and Insights

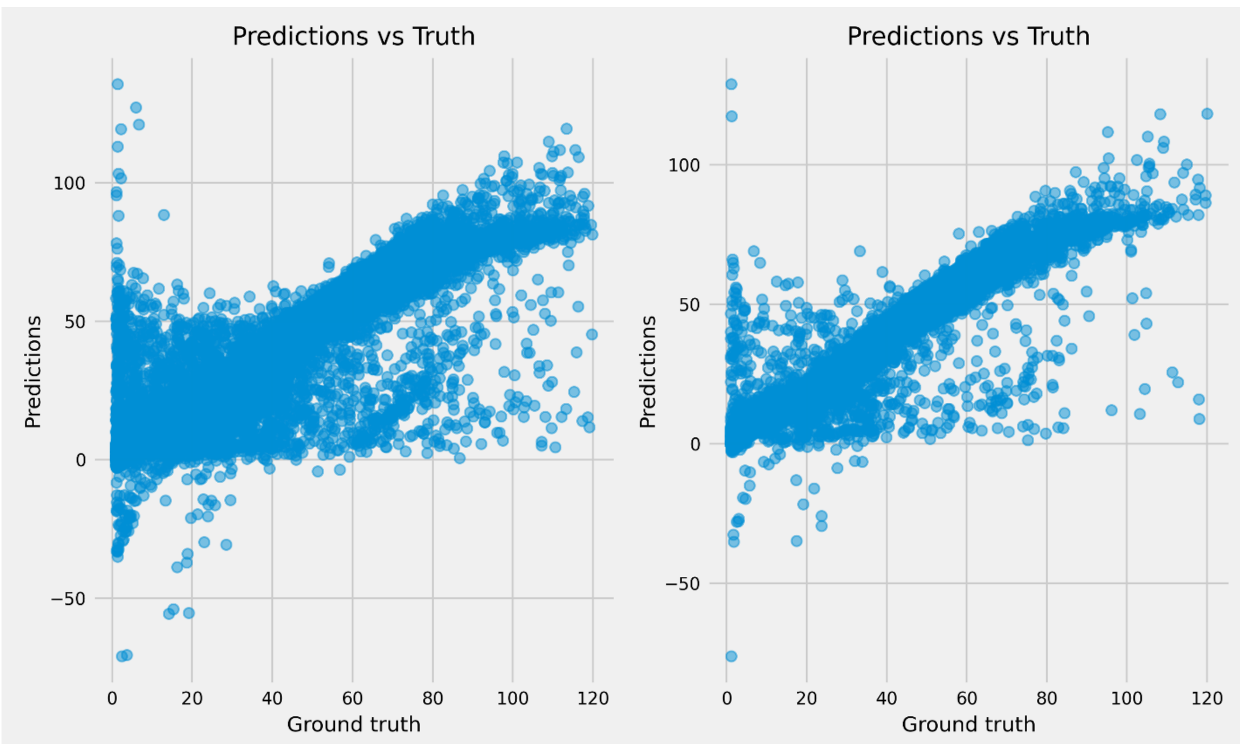
1. The score and rmse of all the model with PCA transformation are very close when compared to the base model with PCA transformation.

2. Without PCA, the models were trained with 92 features compared to 72 features when PCA was applied which means a 13% decrease in the number of features required.

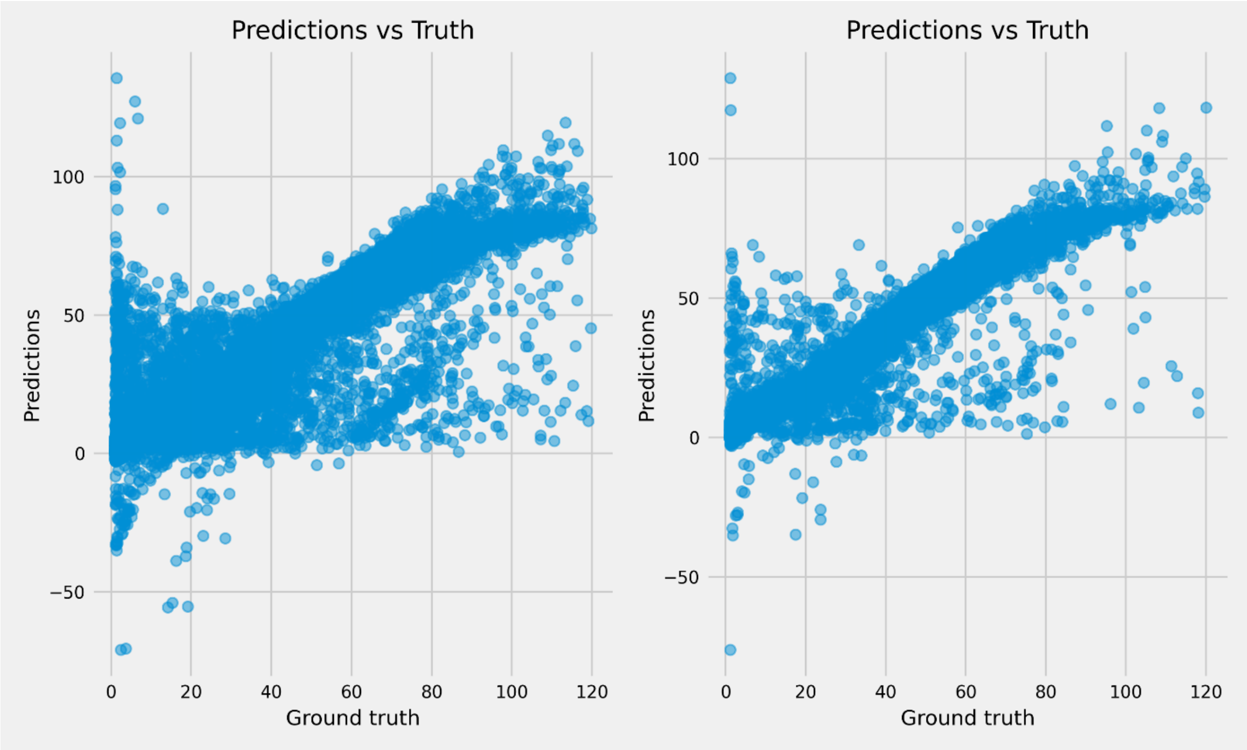
Model Selection

Since the score and rmse were too close, we decided to visualize and compare the plot of predicted vs true values for each model for both training and test data.

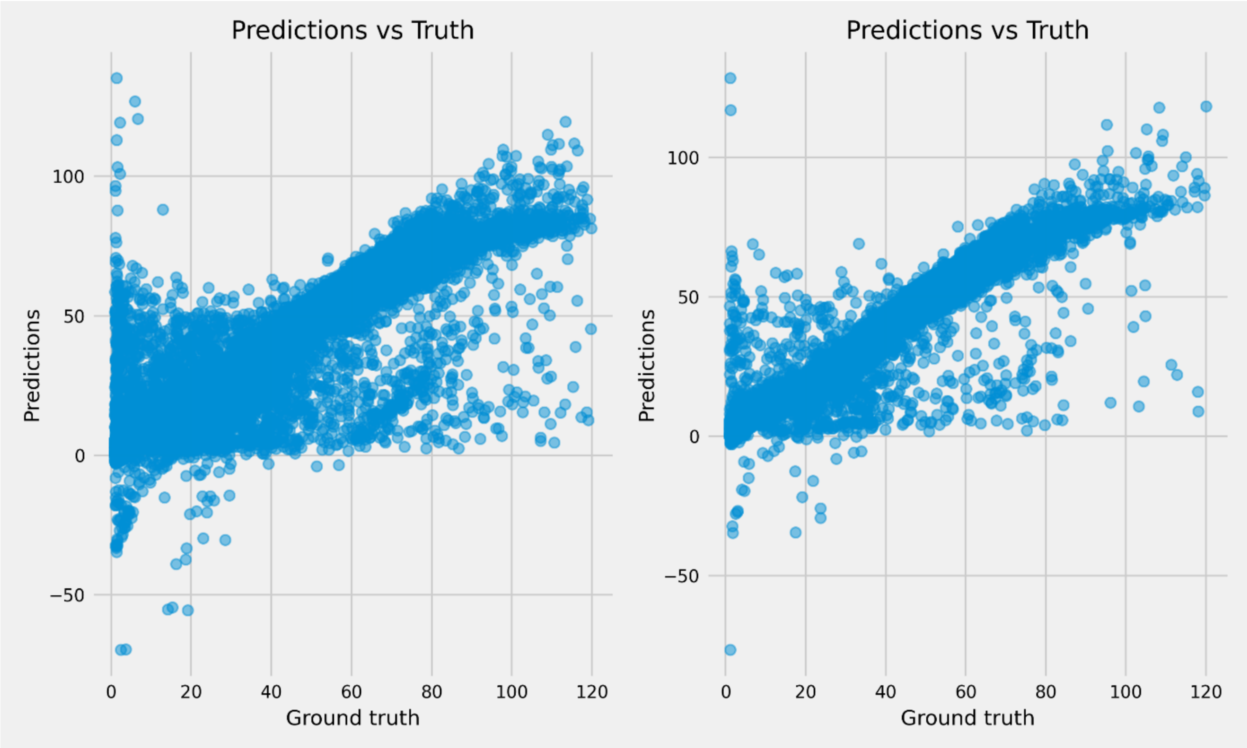
Base Model



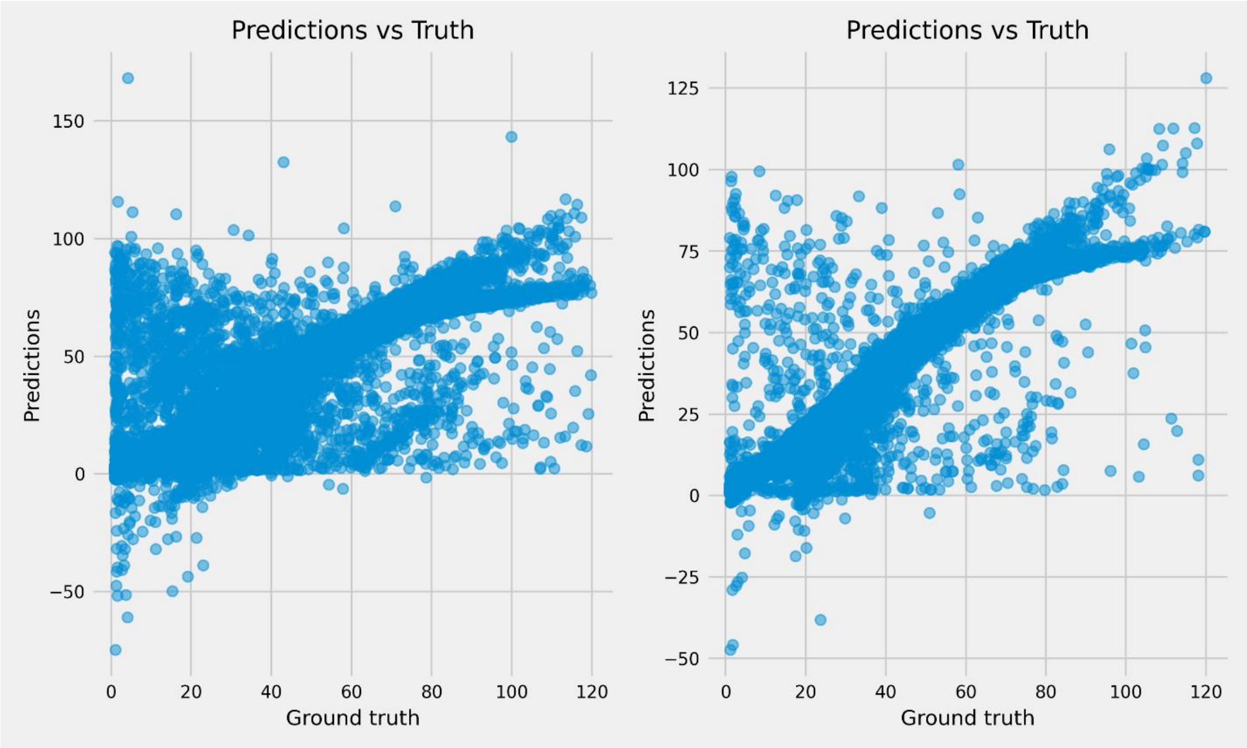
Linear Regression Model



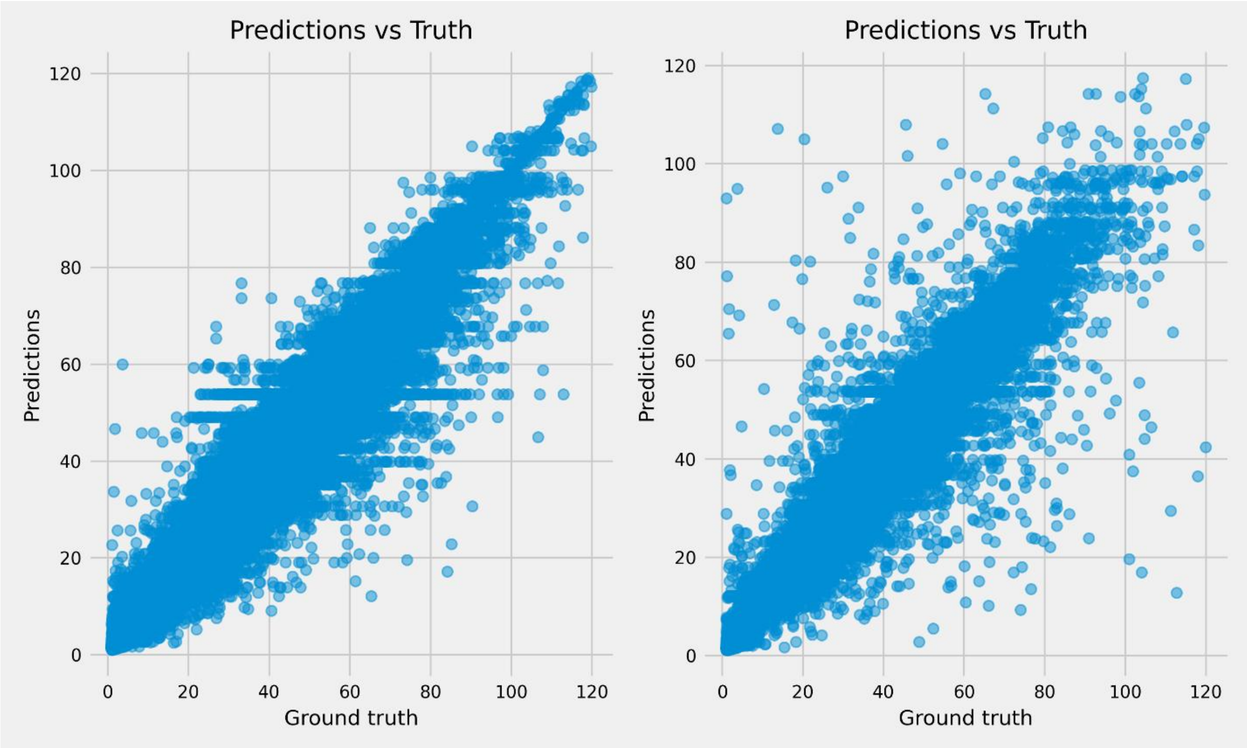
Ridge



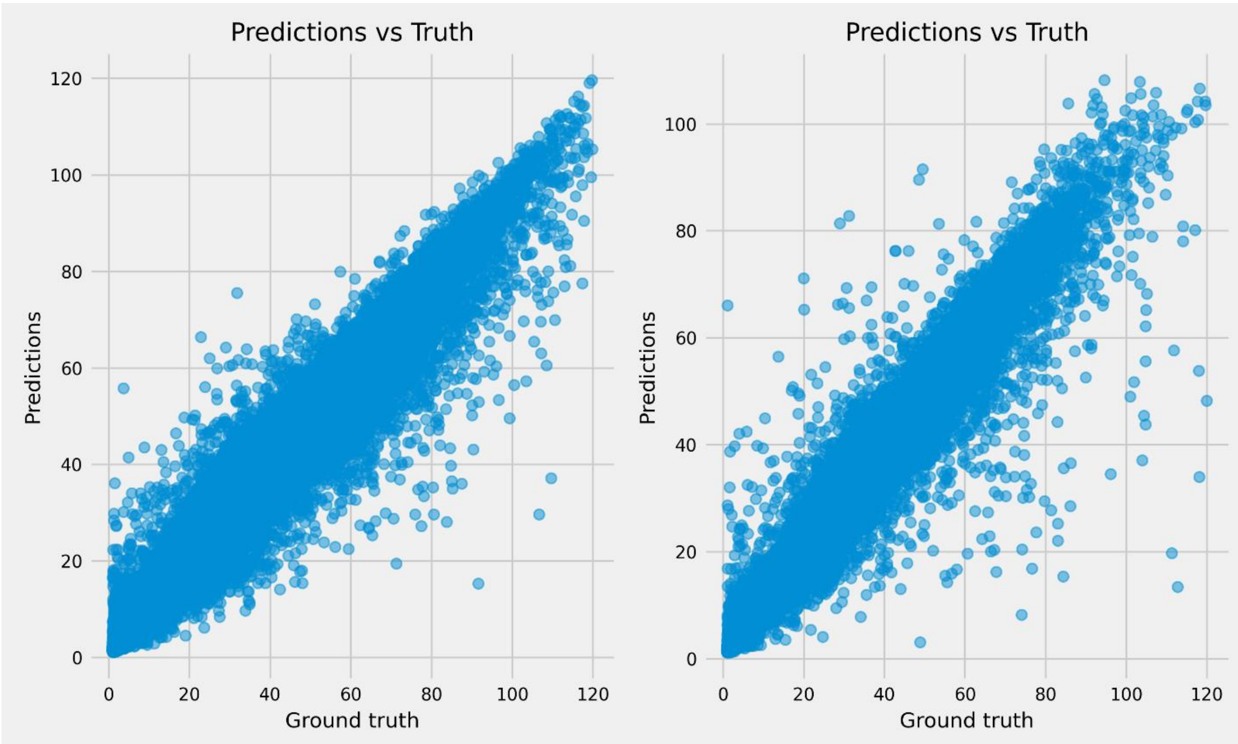
LinearSVR



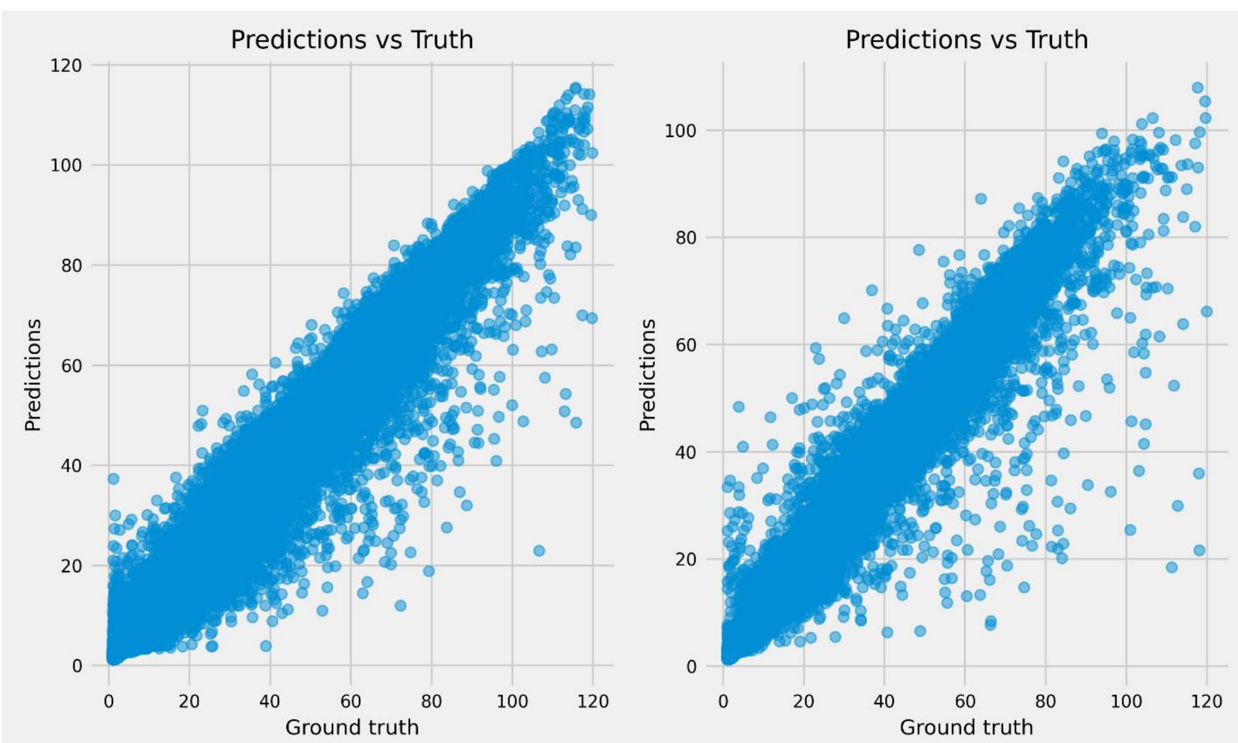
Decision Tree



Random Forest



Extra Tree



We can clearly see from these plots that the four linear models are predicting negative values which is not desirable. As a result, for the final model, we chose the ***Extra tree regressor*** because it predicted positive values, achieved high score with low rmse and does not overfit the data.

Next Steps in model improvement

We will explore kernel PCA to see if it can perform better than the ordinary PCA we used here.

Summary

With the selected model, the company can be able to predict trip duration with a small error of about1.5 minutes which can be acceptable in real life application.